



# Instituto Politécnico Nacional

## Escuela Superior de Cómputo



Desarrollo de Sistemas Distribuidos.

Tarea 3. Chat multicast.

Nombre: Flores Castro Luis Antonio.

Profesor: Pineda Guerrero Carlos.

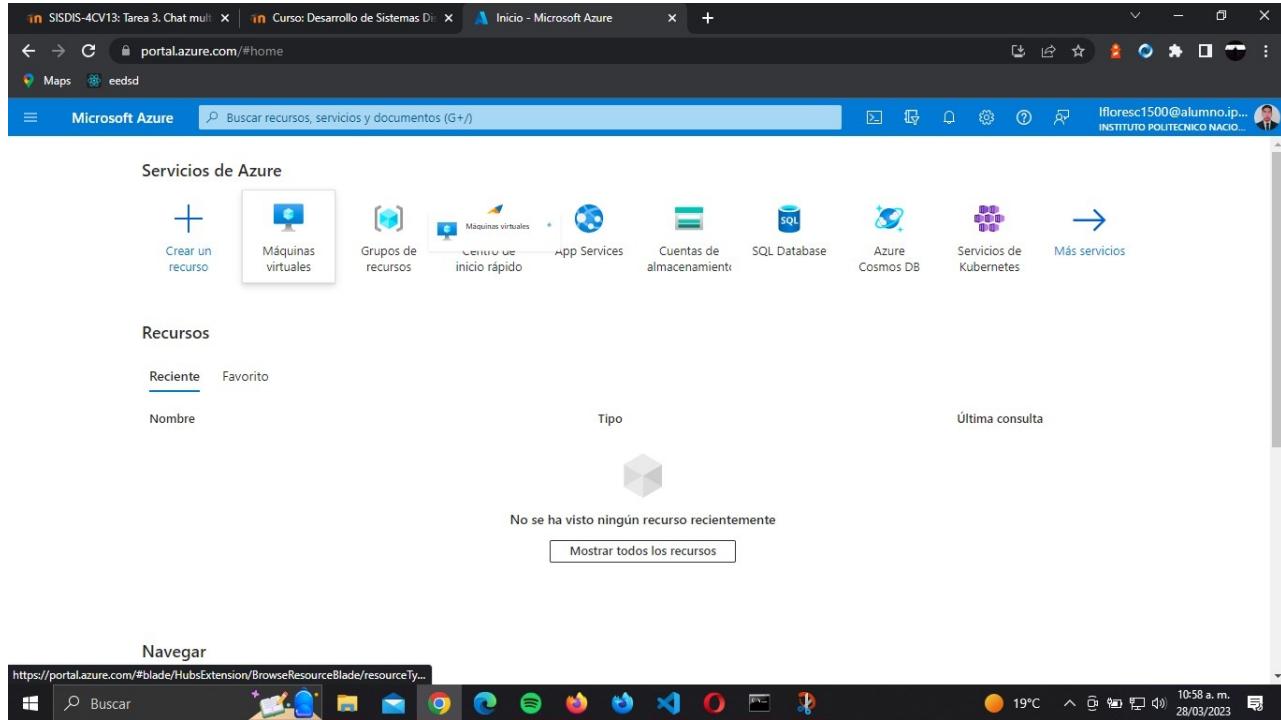
Grupo: 4CV13.

29 Marzo 2023

## Desarrollo.

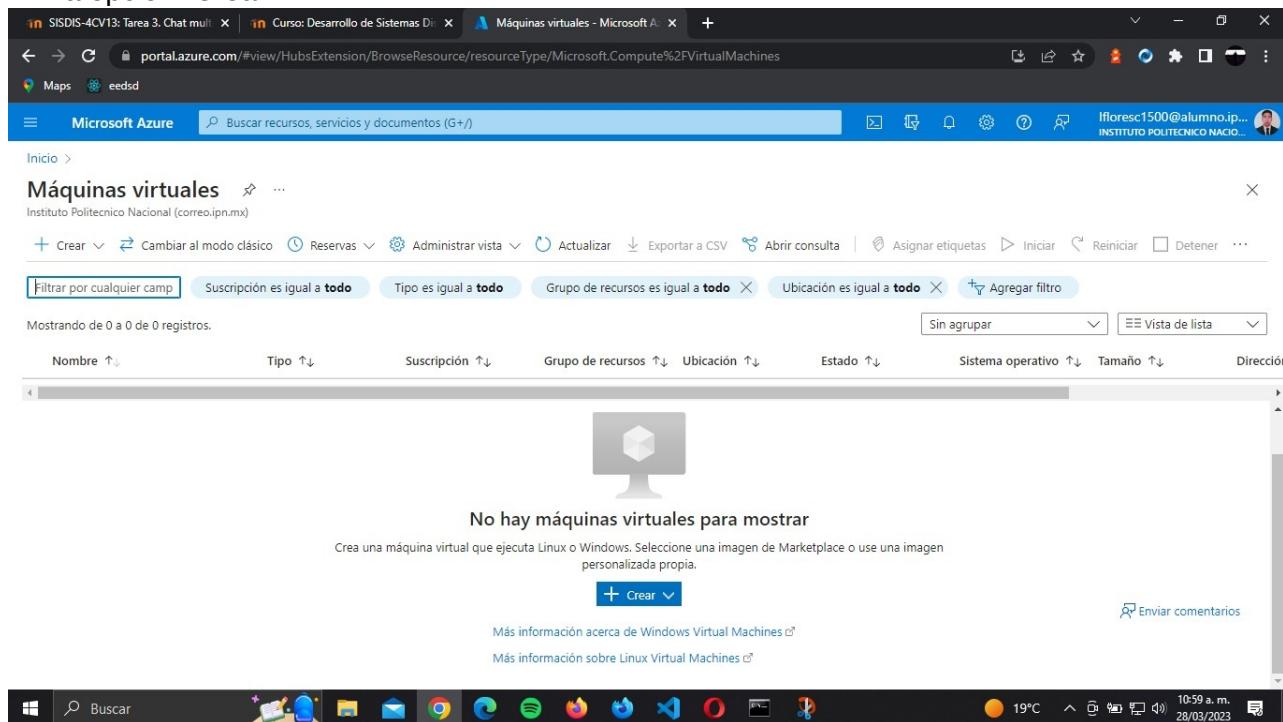
### Creación de la máquina virtual con Windows Server 2012 en Azure.

Para la creación de la primera máquina virtual, primero debemos seleccionar la opción “Máquinas Virtuales” como se muestra en la figura 1.



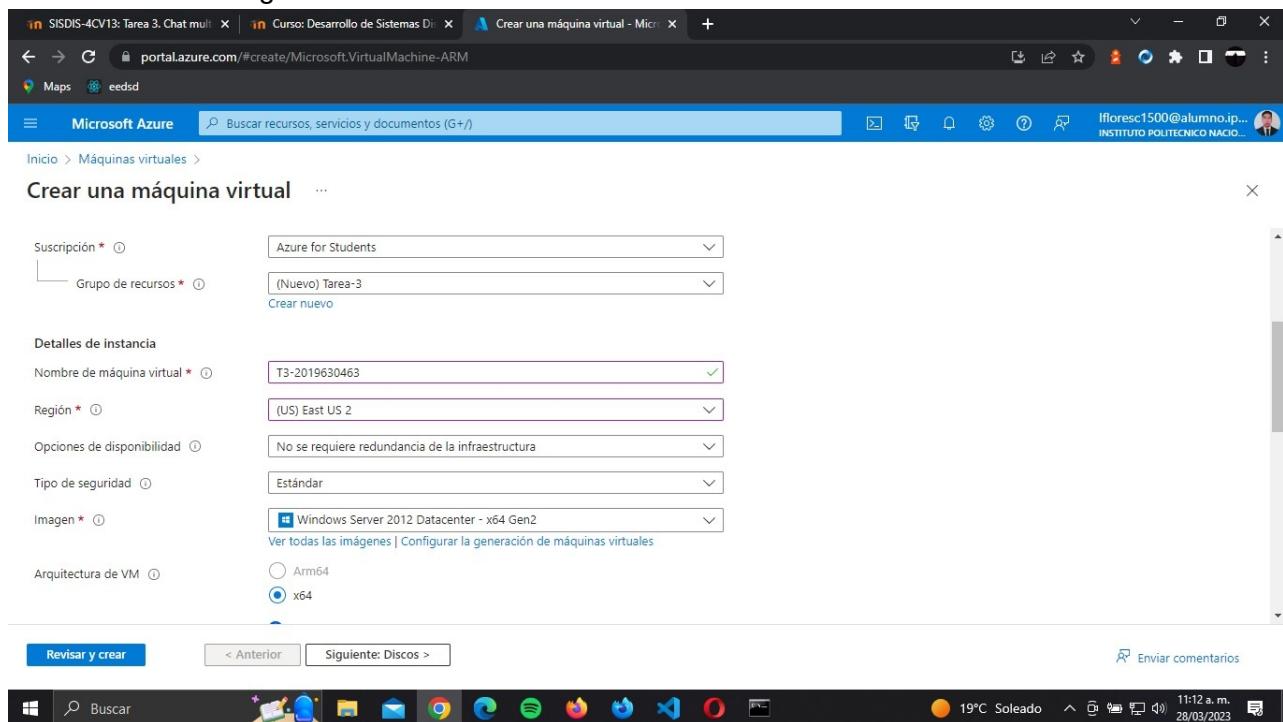
**Figura 1.** Opción *Máquinas Virtuales*.

Se desplegará una pantalla como se muestra en la figura 1.1, aquí debemos dar click en la opción “Crear”.



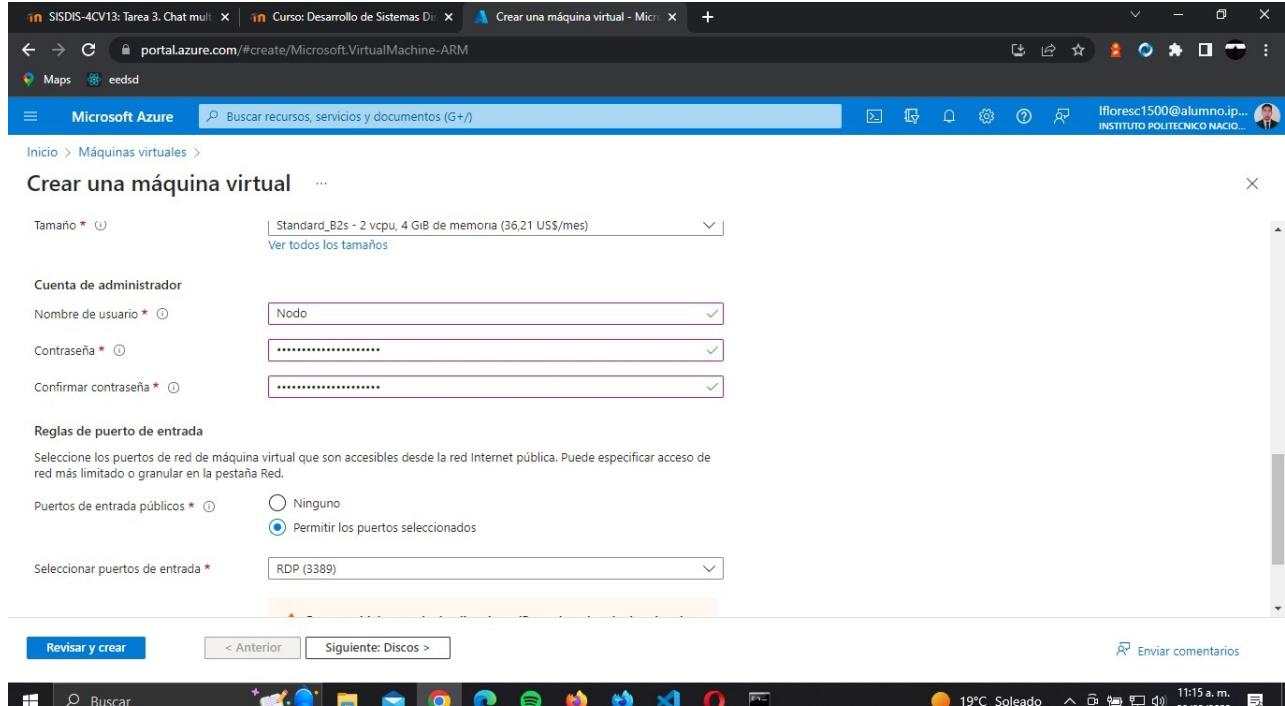
**Figura 1.1. Opción “Crear”.**

Una vez en la pestaña de datos básicos, se creó un grupo de recursos llamado “Tarea-3”, se asignó el nombre “T3-2019630463” a la máquina virtual y se eligió Windows Server 2012 como imagen de sistema.



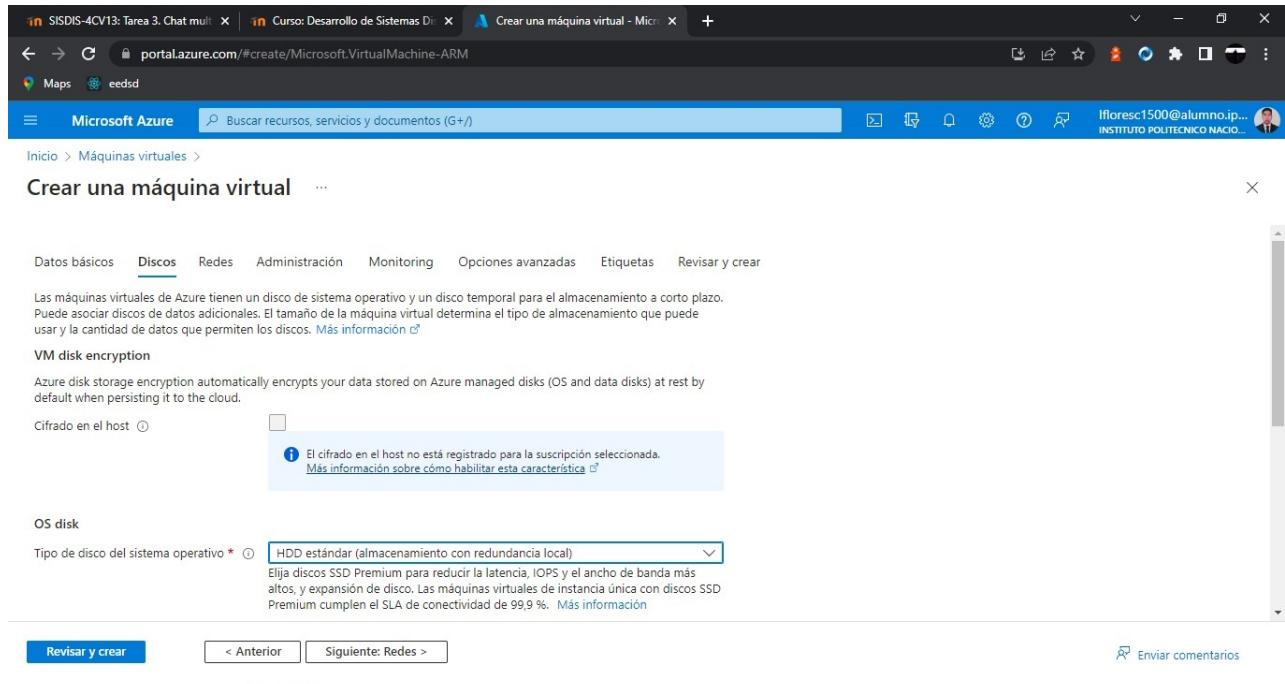
**Figura 1.2. Configuración de datos básicos de máquina virtual.**

Siguiendo con la configuración de datos básicos, ahora se asigna un nombre de usuario y contraseña, el tamaño de disco se elige de 4GiB para que no haya problema al correr la máquina virtual, a su vez se selecciona el puerto de entrada como RDP (3389)



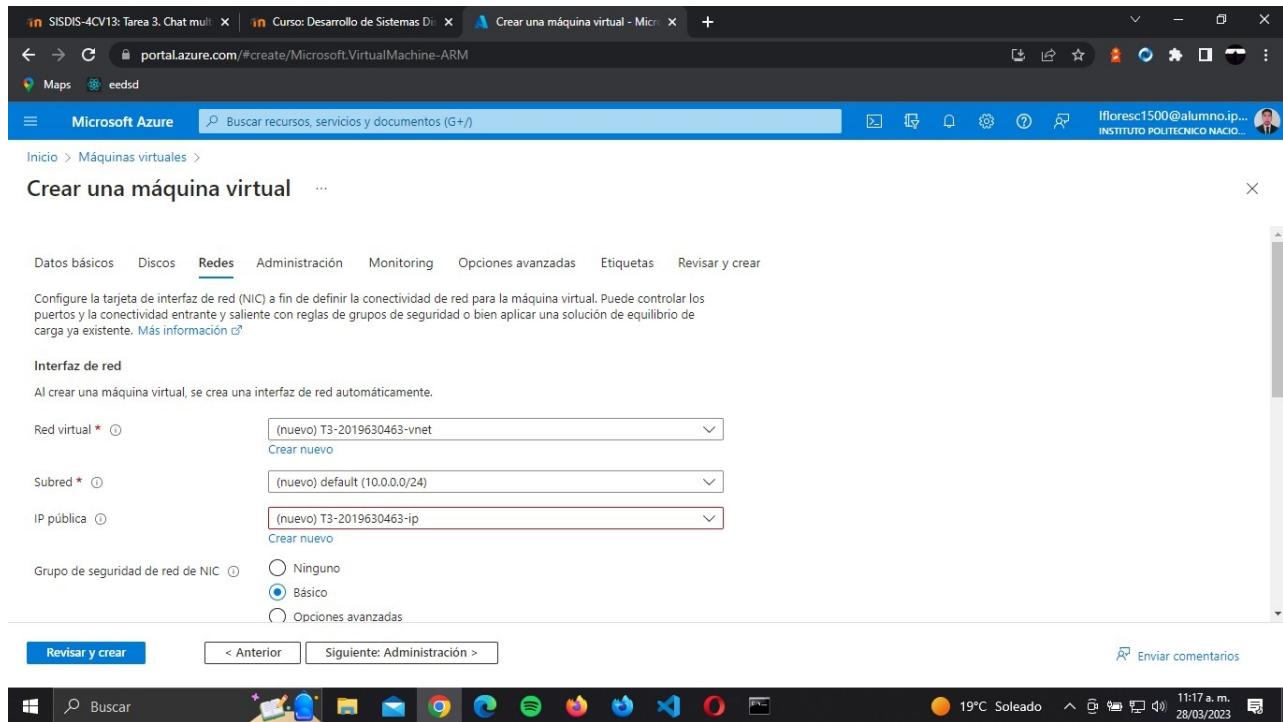
**Figura 1.3.** Configuración de datos básicos de máquina virtual.

Dentro del menú Discos, se selecciona un disco HDD estándar, esto para no generar un gasto mayor en saldo en la cuenta de azure.



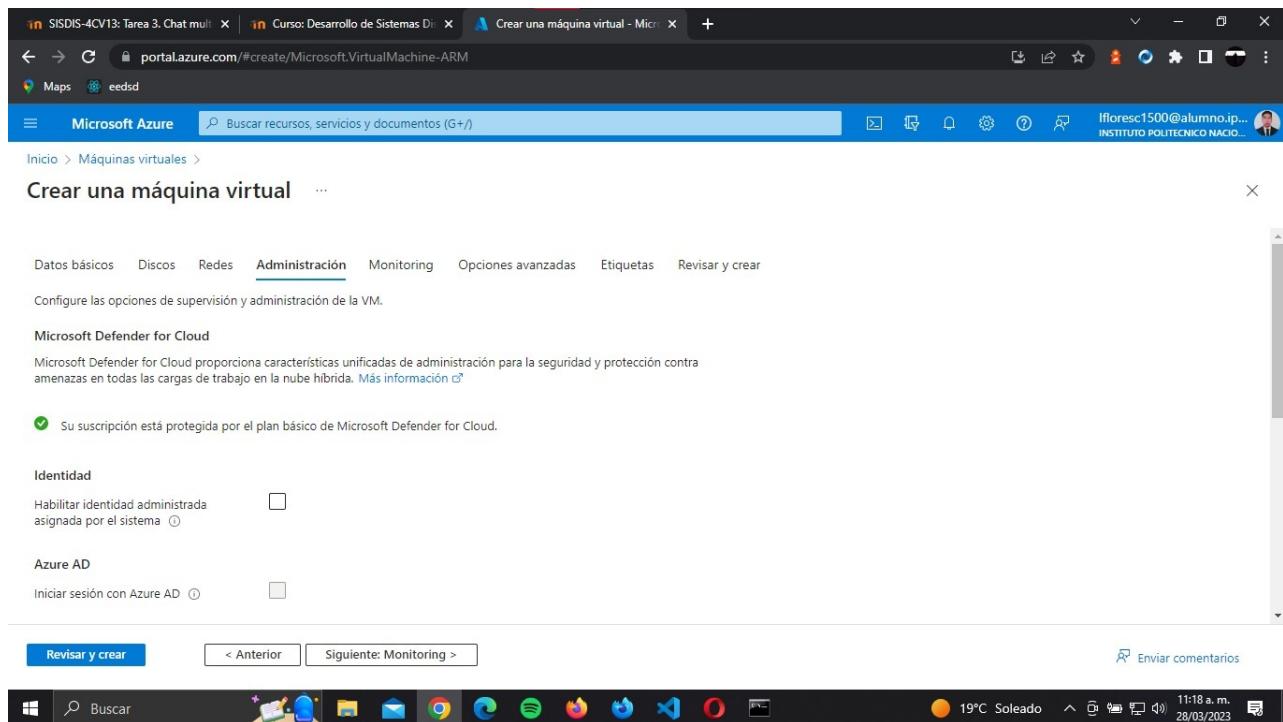
**Figura 1.3.** Configuración de discos de máquina virtual.

Pasando al menú de “Redes”, dejamos la configuración como se encuentra por default.



**Figura 1.4.** Configuración de redes de máquina virtual.

En la parte de “Administración” de igual forma, mantenemos la configuración inicial.



**Figura 1.5.** Configuración administración de máquina virtual.

En el menú “Monitoring” conservamos la configuración que se presenta por default.

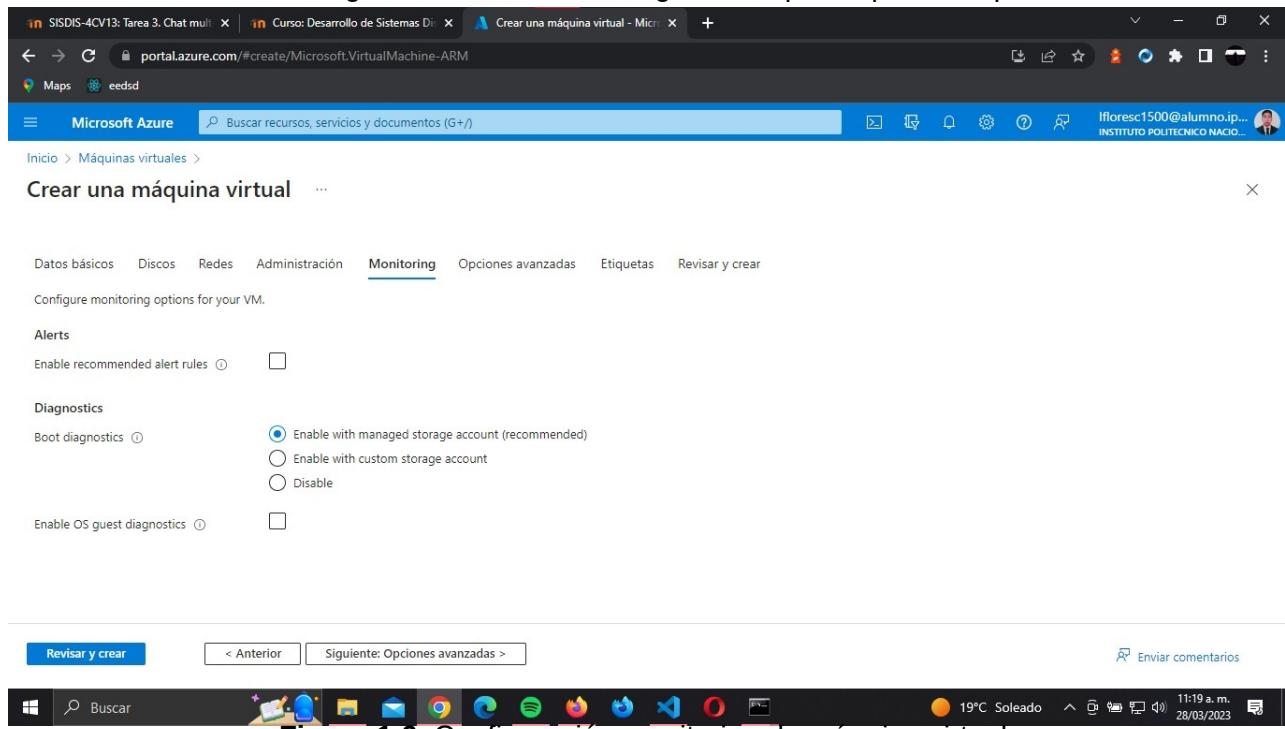


Figura 1.6. Configuración monitoring de máquina virtual.

En la parte de “Opciones avanzadas” de igual manera se conserva la configuración por default.

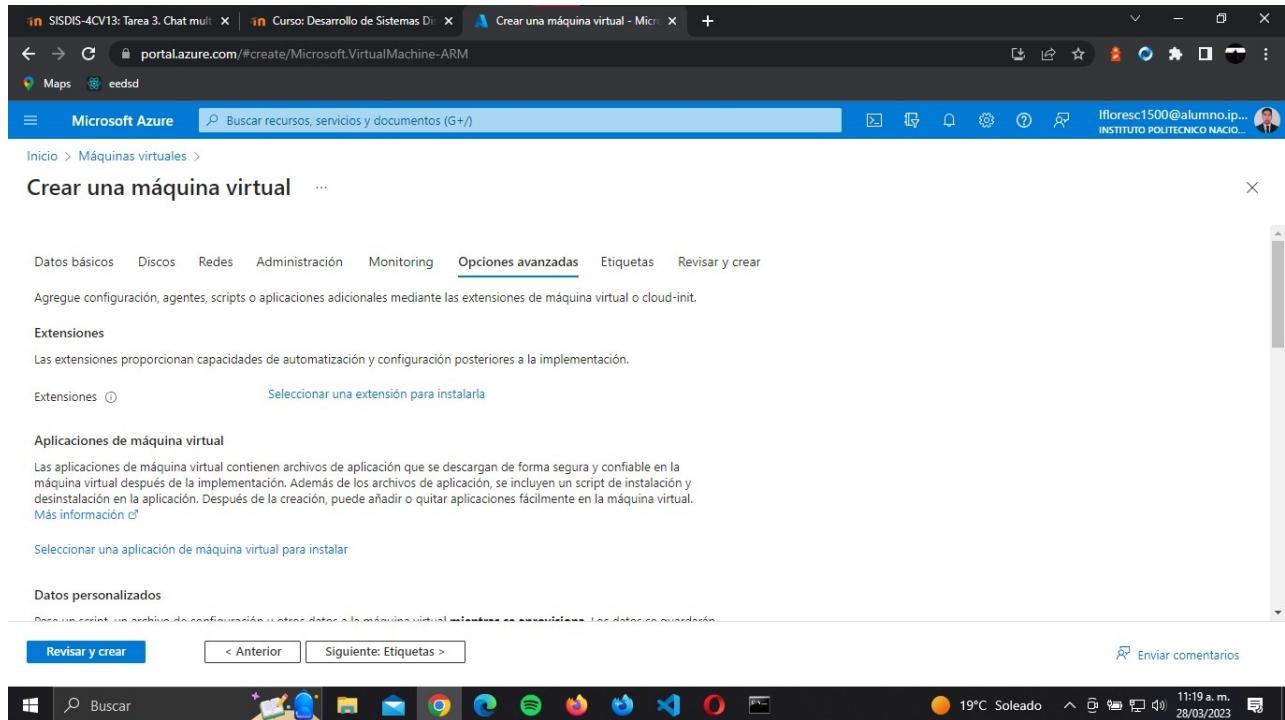


Figura 1.7. Configuración opciones avanzadas de máquina virtual.

En el menú “Etiquetas” dejamos la configuración por default tal y como esta.

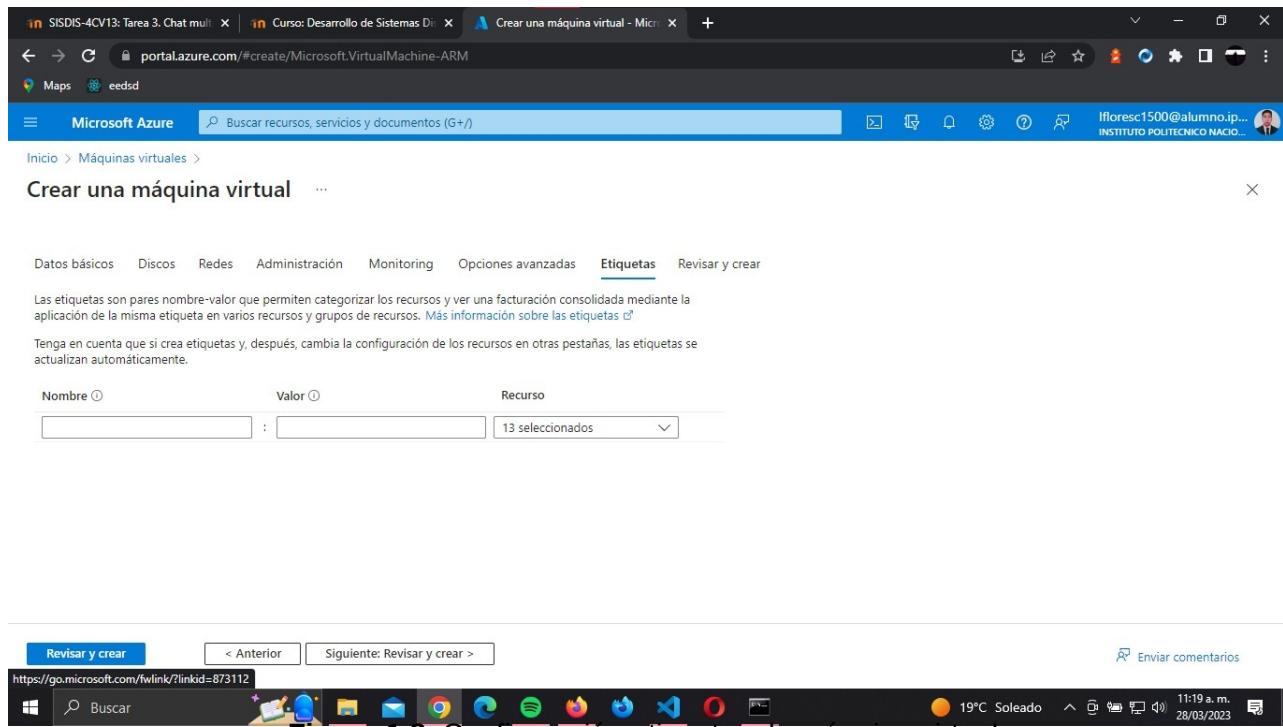


Figura 1.8. Configuración etiquetas de maquina virtual.

Una vez configurado todo, ahora dentro de revisar y crear, nos aparece el resumen de configuración de la máquina virtual, así que se procede a crearla.

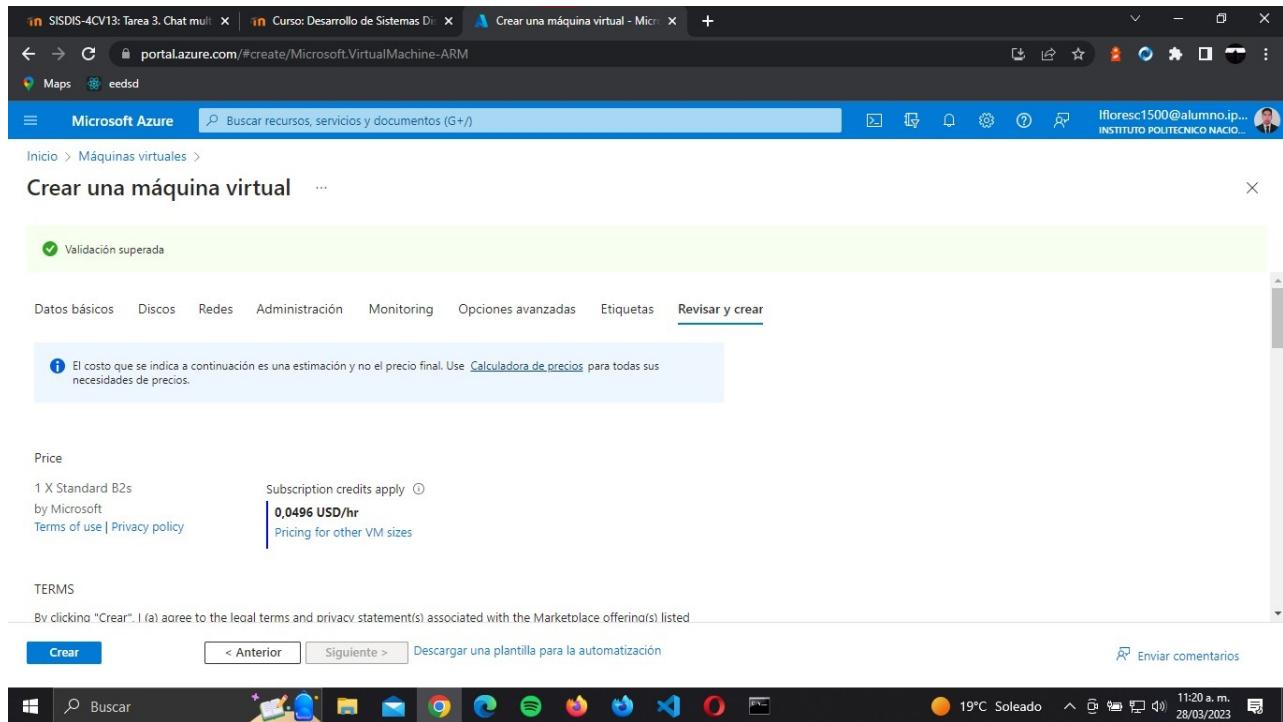


Figura 1.9. Resumen de configuración de máquina virtual.

En la figura 2 se muestra que la creación de la máquina virtual este en proceso.

Screenshot of the Microsoft Azure portal showing the deployment status of 'CreateVm-MicrosoftWindowsServer.WindowsServer-201-20230328111414'. The status is 'La implementación está en curso' (Deployment is in progress). Deployment details:

| Recurso            | Tipo                         | Estado | Detalles de la operación |
|--------------------|------------------------------|--------|--------------------------|
| T3-2019630463-vnet | Microsoft.Network/virtual... | OK     | Detalles de la operación |
| T3-2019630463-nsg  | Microsoft.Network/network... | OK     | Detalles de la operación |
| T3-2019630463-ip   | Microsoft.Network/public...  | OK     | Detalles de la operación |

Figura 2. Resumen de configuración de máquina virtual.

En la figura 2.1, se muestra que la máquina virtual ha sido creada de forma exitosa.

Screenshot of the Microsoft Azure portal showing the completed deployment status of 'CreateVm-MicrosoftWindowsServer.WindowsServer-201-20230328111414'. The status is 'Se completó la implementación' (Deployment completed). Deployment details:

| Recurso            | Tipo                         | Estado | Detalles de la operación |
|--------------------|------------------------------|--------|--------------------------|
| T3-2019630463-vnet | Microsoft.Network/virtual... | OK     | Detalles de la operación |
| T3-2019630463-nsg  | Microsoft.Network/network... | OK     | Detalles de la operación |
| T3-2019630463-ip   | Microsoft.Network/public...  | OK     | Detalles de la operación |

Figura 2.1. Creación de máquina virtual finalizada.

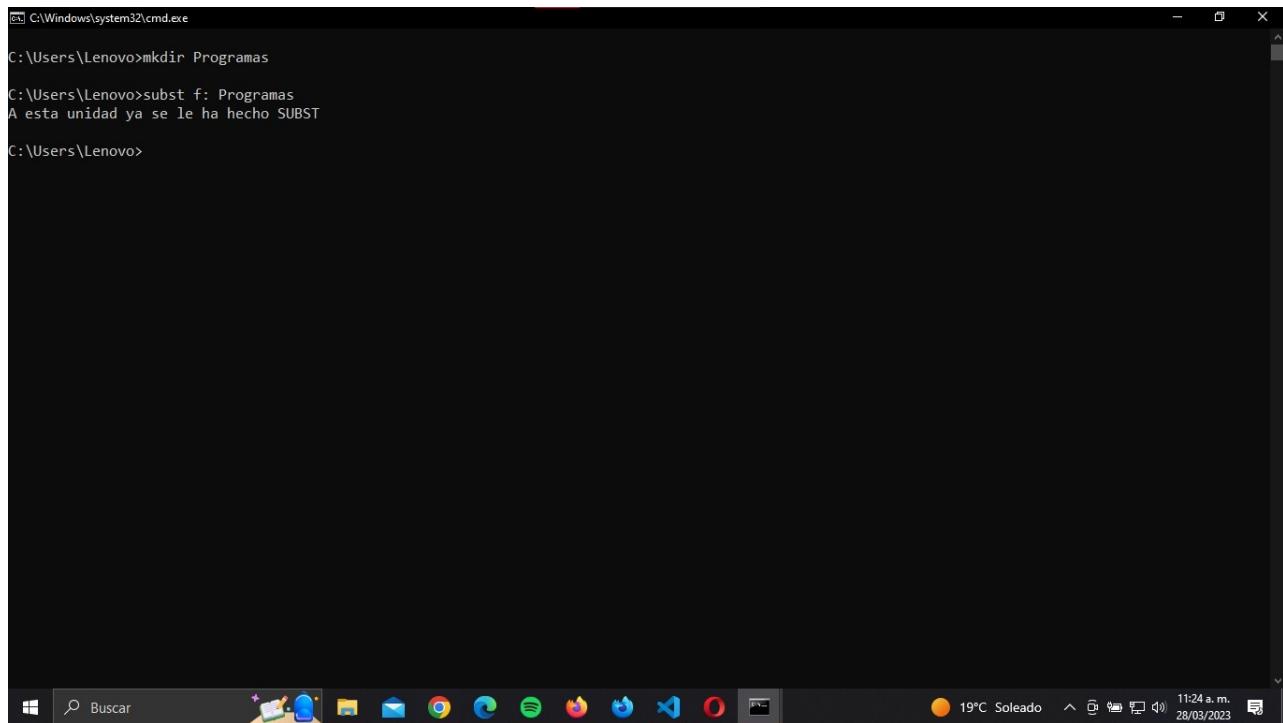
Se despliega la información general de la máquina virtual, donde se muestra el nombre, dirección IP pública, entre otra información de esta.

Figura 2.2. Información general de la máquina virtual.

Una vez creada y configurada la máquina virtual procederemos a realizar la conexión de nuestra PC a esta, nos dirigimos al menú “Conectar” y seleccionamos el botón “Descargar archivo RDP”.

Figura 2.3. Descarga de archivo RDP de máquina virtual.

Ya descargado el archivo RDP, ahora crearemos un disco lógico para poder interactuar con los archivos que se encuentren en nuestra PC y la máquina virtual, así que abrimos una terminal y creamos una carpeta llamada “Programas” con el comando `mkdir Programas`, una vez creada ahora ejecutamos el comando `subst f: Programas`, el cual va permitir asociar una ruta de acceso con la letra f.



```
C:\Windows\system32\cmd.exe
C:\Users\Lenovo>mkdir Programas
C:\Users\Lenovo>subst f: Programas
A esta unidad ya se le ha hecho SUBST
C:\Users\Lenovo>
```

**Figura 2.4.** Descarga de archivo RDP de máquina virtual.

Abriendo el explorador de archivos, podemos ver que el disco lógico “F” se ha creado.

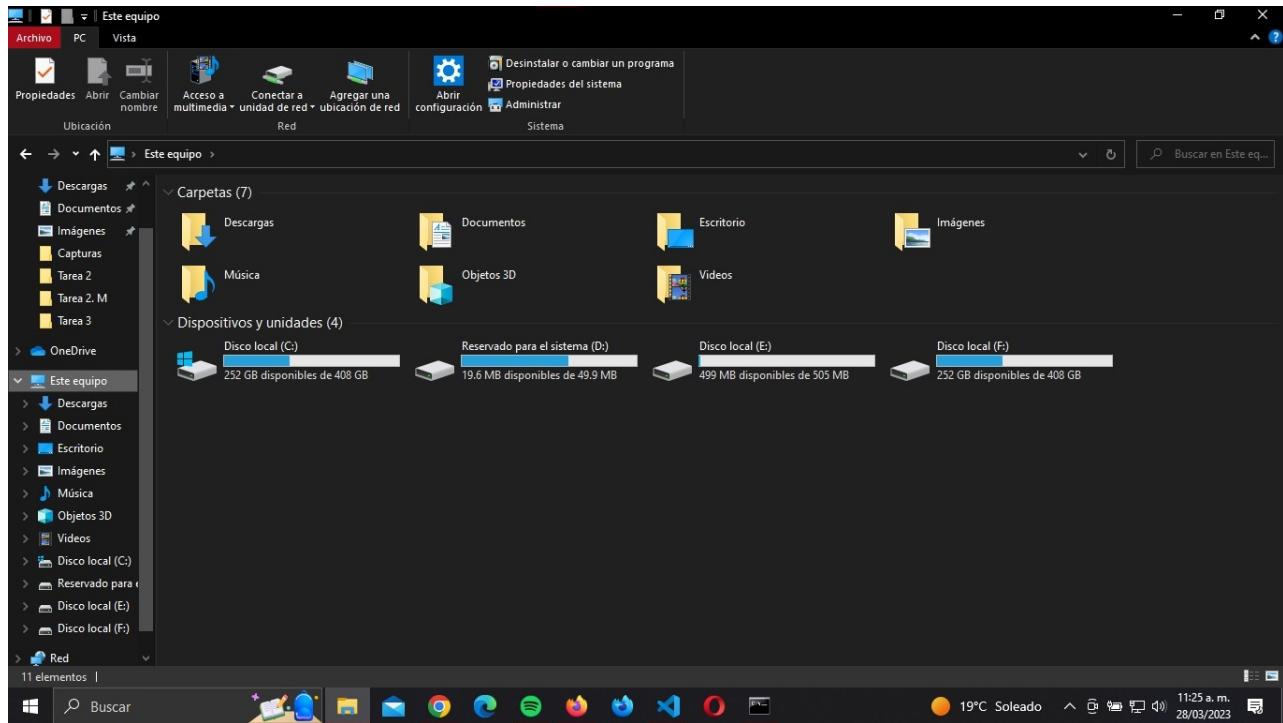


Figura 2.5. Creación de disco lógico.

Después damos click derecho sobre el archivo RDP que se había descargado y seleccionamos la opción editar y se desplegará la siguiente imagen, allí seleccionaremos la opción “Más”.

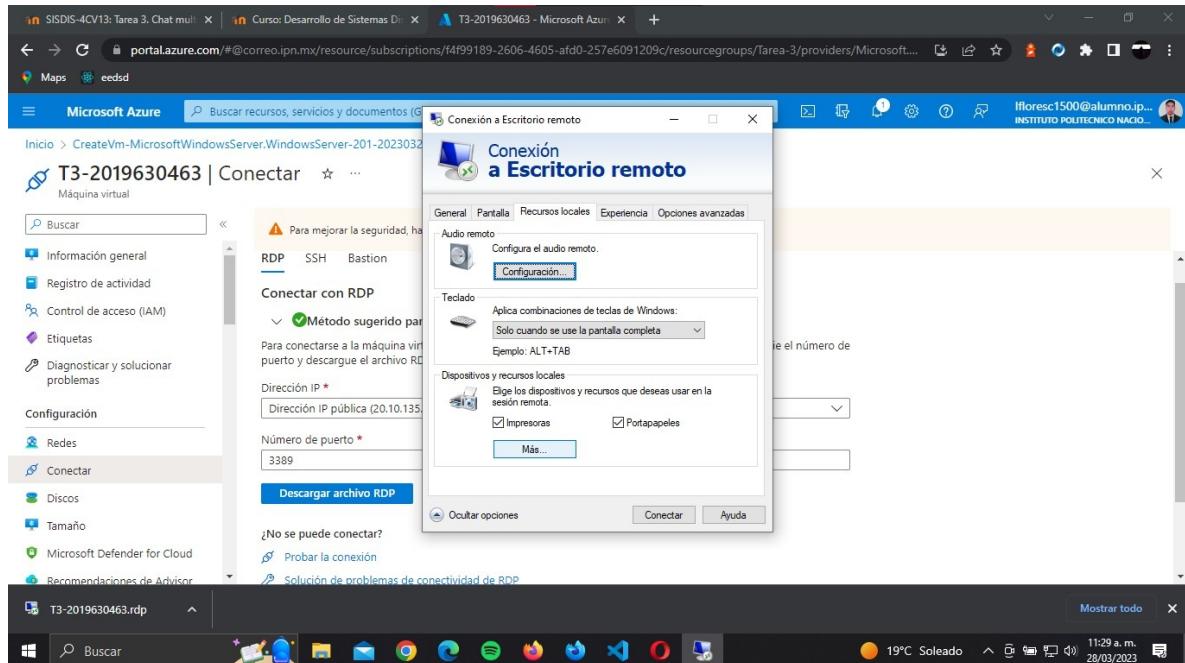


Figura 2.6. Configuración de archivo RDP.

Allí se seleccionaremos el menú unidades y seleccionaremos el disco local “F” y aceptamos.

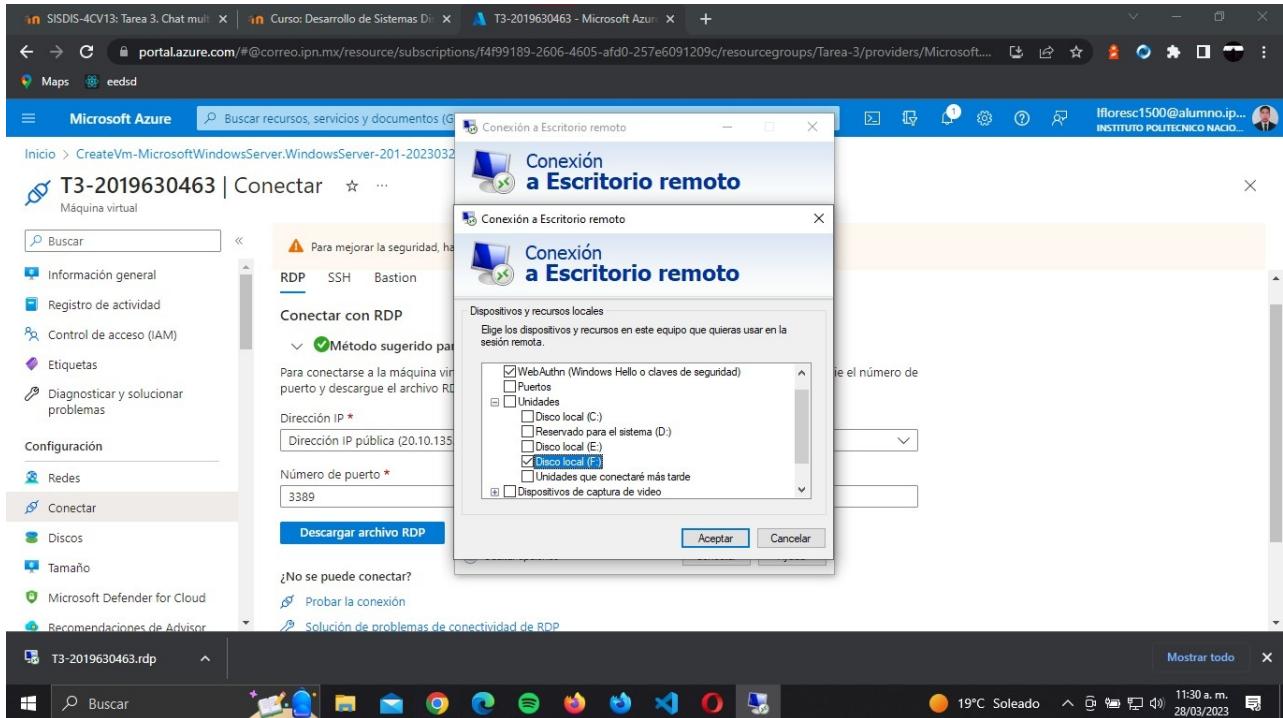


Figura 2.7. Configuración de archivo RDP.

Ya seleccionado el disco ahora damos click en conectar.

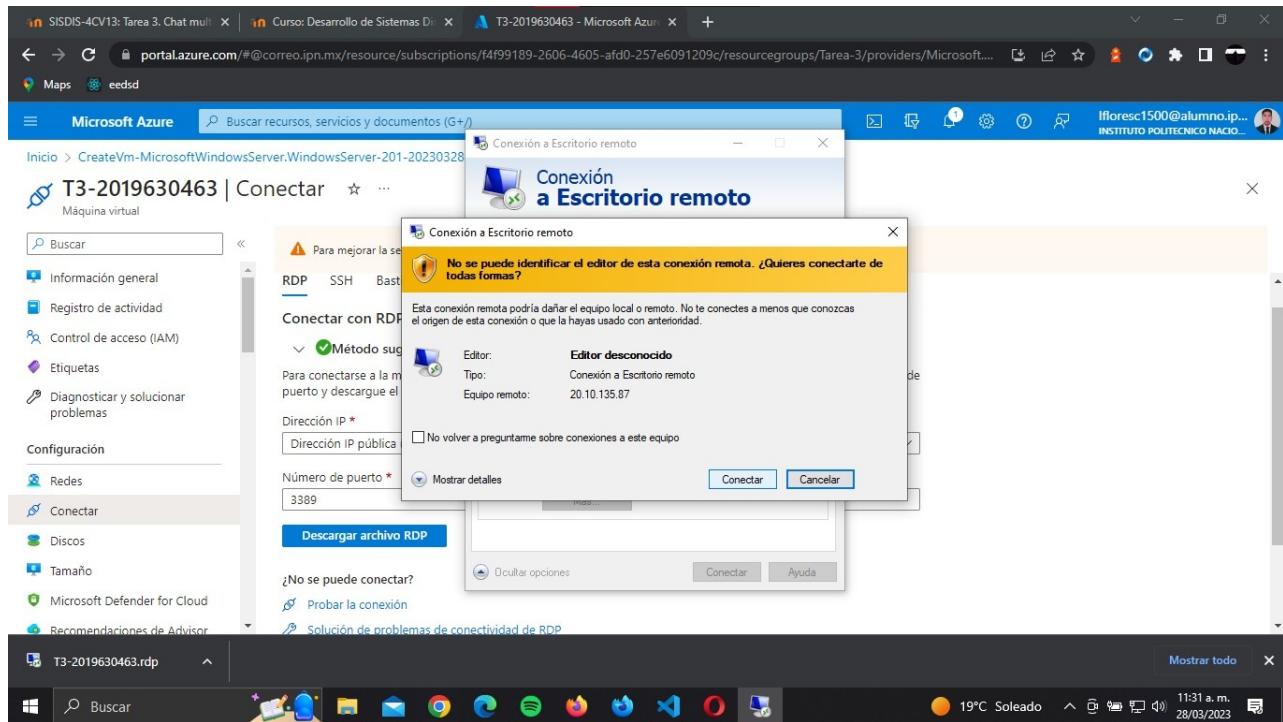


Figura 2.8. Conexión a máquina virtual.

Después de seleccionar conectar, ahora se nos desplegará una ventana, donde se indica que introduzcamos el usuario y contraseña de la máquina virtual.

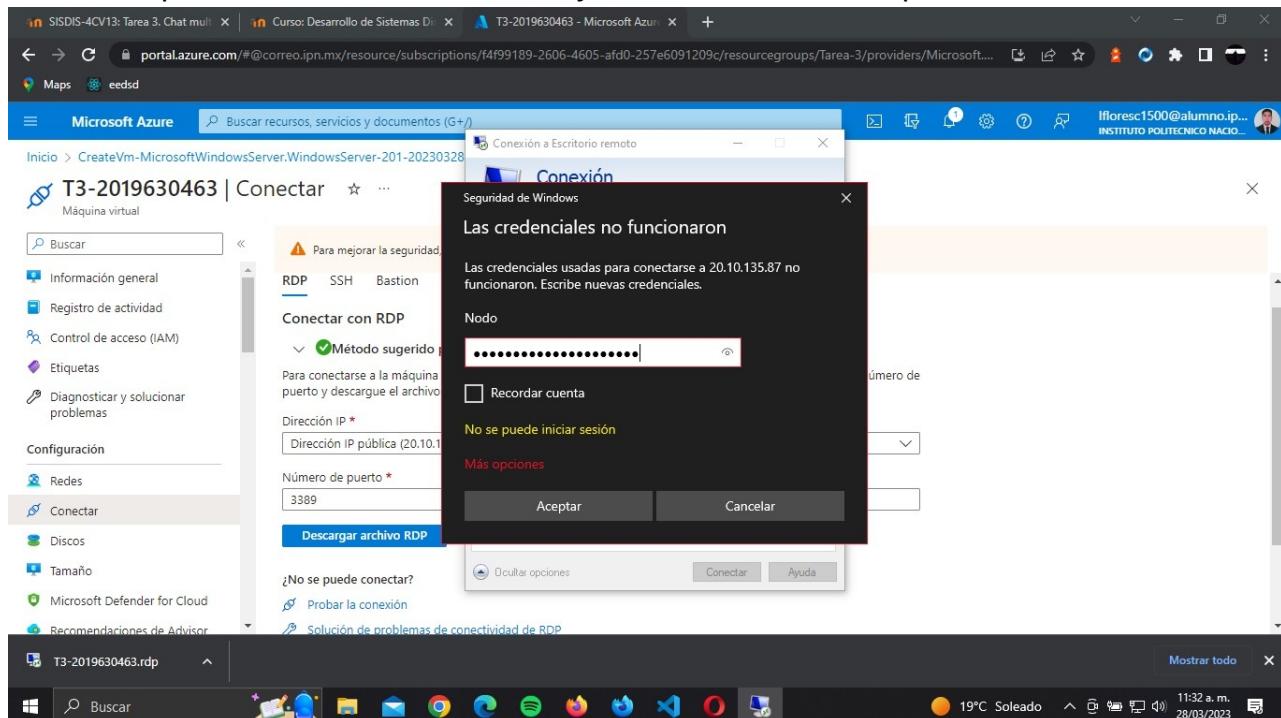


Figura 2.9. Ingreso de credenciales para conexión de máquinas virtuales.

Después de ingresar las credenciales, aceptamos la conexión.

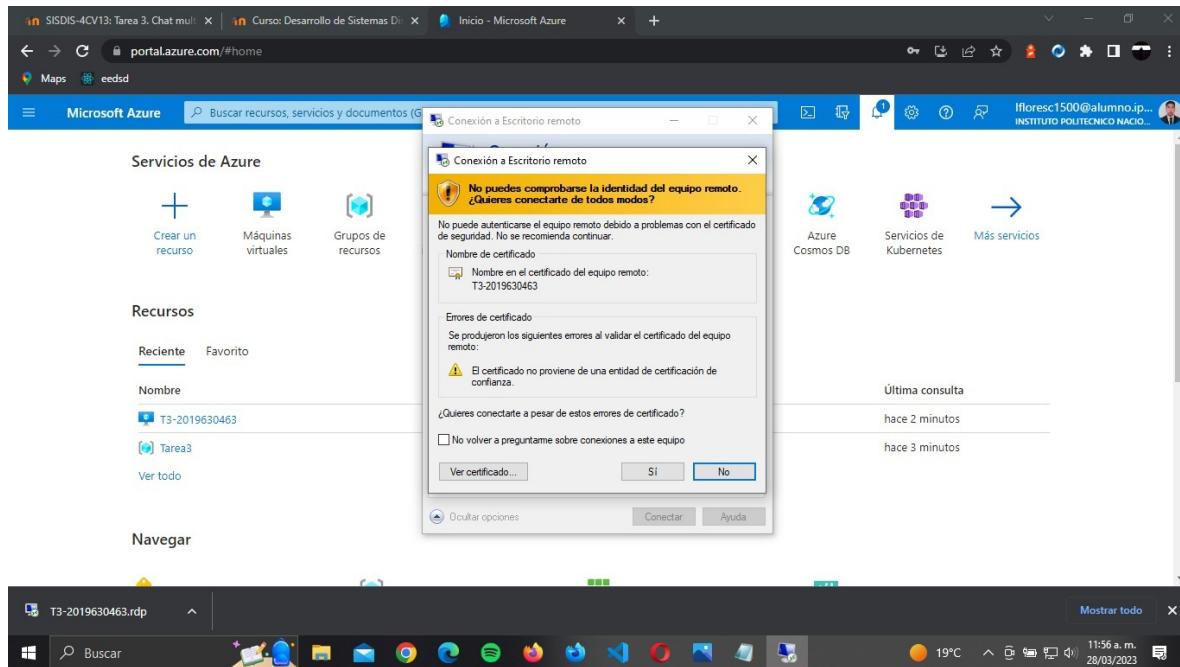


Figura 3. Conexión a máquina virtual.

Ya aceptada la conexión, ahora se inicia la máquina virtual.

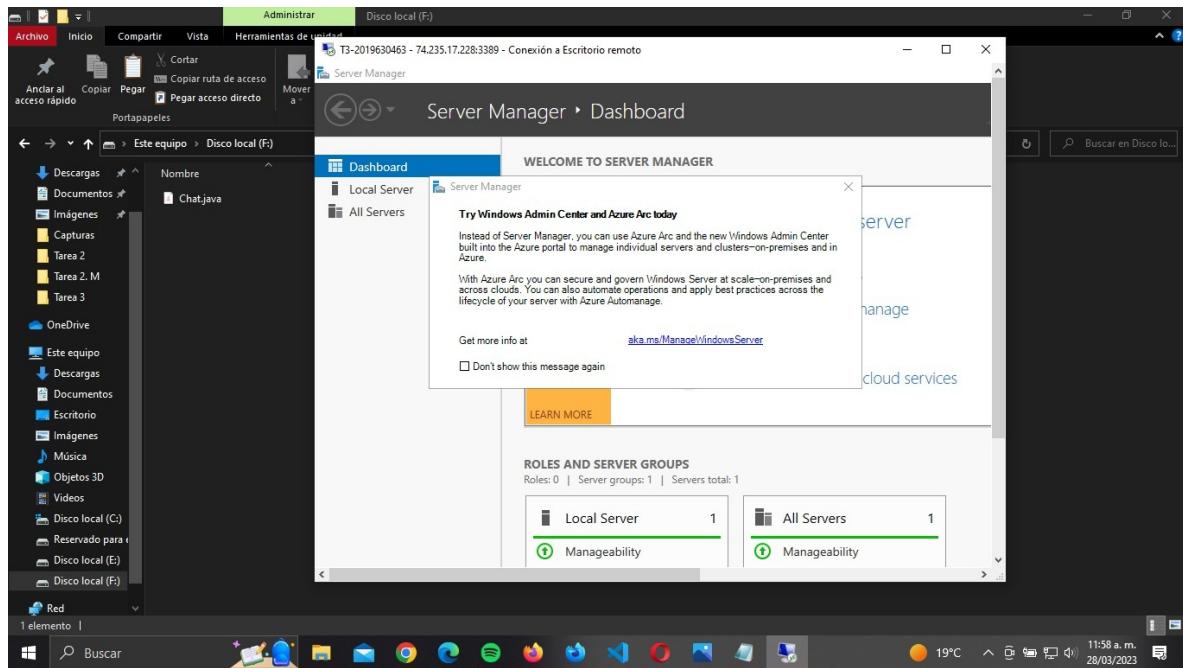


Figura 3.1 Inicio de sesión de la máquina virtual.

Se realiza la instalación de JDK en la máquina virtual para la compilación del programa.

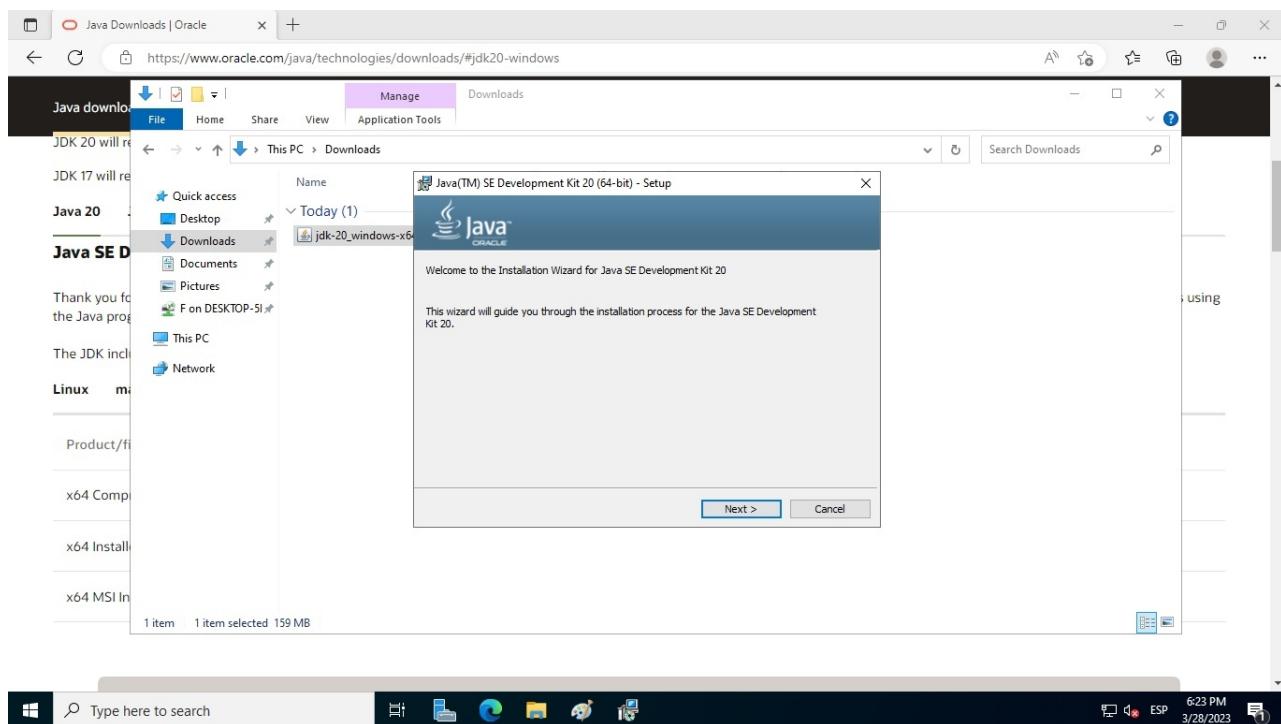
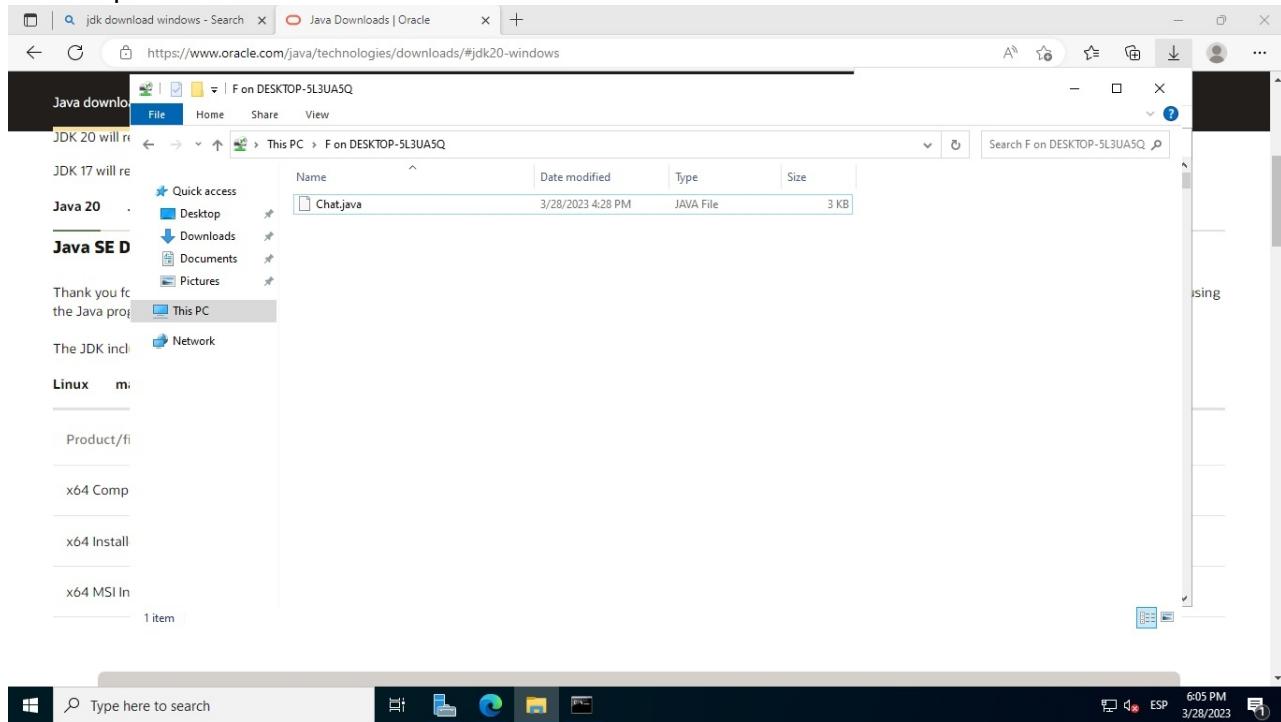


Figura 3.2 Instalación de JDK en máquina virtual.

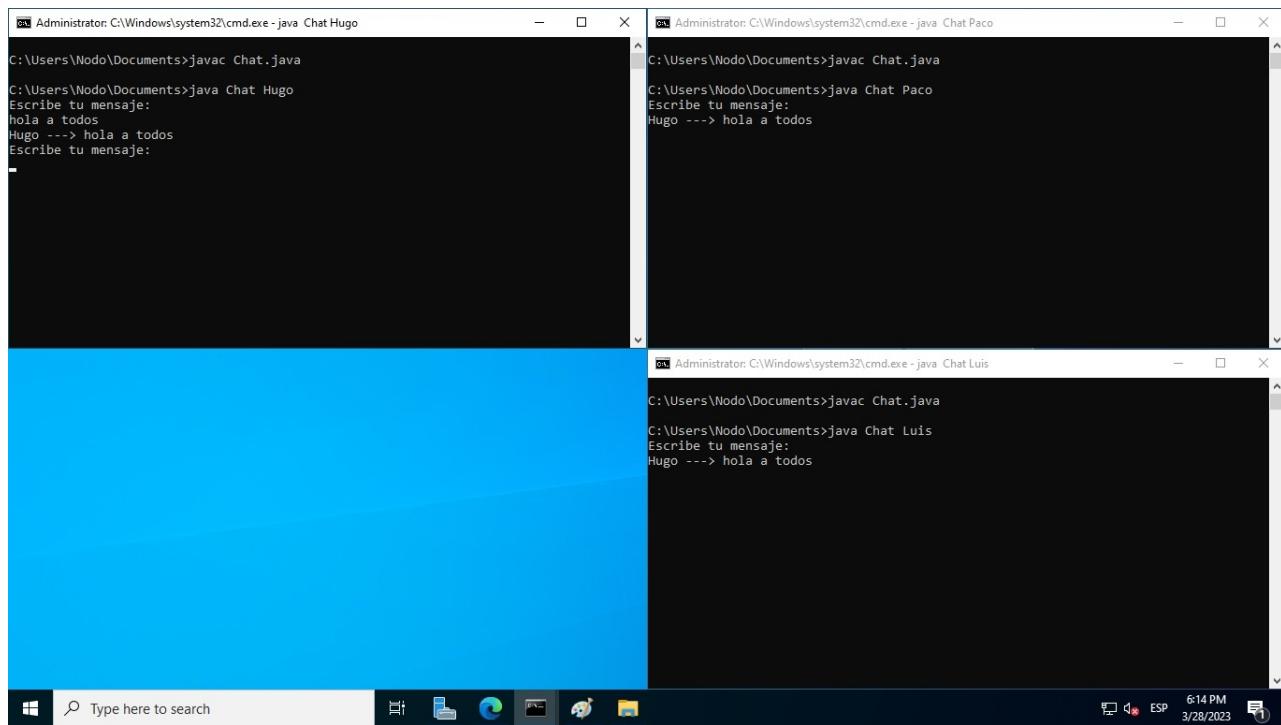
Nos dirigimos al explorador de archivos de la máquina virtual y seleccionamos el disco lógico "F" que se había creado en nuestra PC y podemos ver el archivo java que se compartió anteriormente desde nuestra PC.



**Figura 3.3** Visualización de archivos compartidos mediante disco lógico entre PC local y máquina virtual.

Ahora abrimos 3 terminales en la máquina virtual, compilamos el programa “Chat.java” en cada una de ellas y ejecutamos el programa pasando como parámetro el nombre de cada usuario, en este caso Hugo, Paco y Luis.

Iniciamos la conversación con el usuario “Hugo” el cual envía el mensaje “*hola a todos*”, este mensaje se muestra en las ventanas de los demás usuarios y en la del usuario que ha enviado el mensaje.



The image shows three separate terminal windows (cmd.exe) running on a Windows operating system. Each window has "Administrator" privileges and is located at C:\Windows\system32\cmd.exe. The first window, titled "Administrator: C:\Windows\system32\cmd.exe - java Chat Hugo", shows the user "Hugo" sending a message to "todos". The second window, titled "Administrator: C:\Windows\system32\cmd.exe - java Chat Paco", shows the user "Paco" receiving the message. The third window, titled "Administrator: C:\Windows\system32\cmd.exe - java Chat Luis", shows the user "Luis" receiving the same message. All three users are prompted to "Escribe tu mensaje:" (Write your message:).

```
C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Hugo
Escribe tu mensaje:
hola a todos
Hugo --> hola a todos
Escribe tu mensaje:

Administrator: C:\Windows\system32\cmd.exe - java Chat Paco
C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Paco
Escribe tu mensaje:
Hugo --> hola a todos

Administrator: C:\Windows\system32\cmd.exe - java Chat Luis
C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Luis
Escribe tu mensaje:
Hugo --> hola a todos
```

**Figura 3.4** Inicio de conversación con el usuario Hugo.

Ahora, Paco envía “*hola Hugo*”.

The screenshot shows three separate Windows Command Prompt windows running under administrator privileges. Each window has a title bar indicating it's an administrator session and the command used to run the Java application (javac Chat.java or java Chat [username]). The windows are arranged vertically on a blue desktop background. The first window (Hugo) shows Hugo sending a message to everyone. The second window (Paco) shows Paco responding to Hugo. The third window (Luis) shows Luis joining the conversation. The system tray at the bottom right shows the date and time as 3/28/2023, 6:15 PM, and icons for battery, signal, and network.

```
C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Hugo
Escribe tu mensaje:
hola a todos
Hugo ---> hola a todos
Escribe tu mensaje:
Paco ---> hola Hugo

C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Paco
Escribe tu mensaje:
Hugo ---> hola a todos
hola Hugo
Paco ---> hola Hugo
Escribe tu mensaje:

C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Luis
Escribe tu mensaje:
Hugo ---> hola a todos
Paco ---> hola Hugo
```

**Figura 3.5.** Paco envía “*hola Hugo*”.

Luis envía “*hola Paco*” y “*hola Hugo*”.

This screenshot shows the same three Windows Command Prompt windows as Figure 3.5. In this sequence, Luis is the active participant, sending messages to Hugo and Paco. The windows are arranged vertically on a blue desktop background. The first window (Hugo) shows Hugo sending a message to everyone. The second window (Paco) shows Paco responding to Hugo. The third window (Luis) shows Luis joining the conversation and taking the lead. The system tray at the bottom right shows the date and time as 3/28/2023, 6:16 PM, and icons for battery, signal, and network.

```
C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Hugo
Escribe tu mensaje:
hola a todos
Hugo ---> hola a todos
Escribe tu mensaje:
Paco ---> hola Hugo
Luis ---> hola Paco
Luis ---> hola Hugo

C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Paco
Escribe tu mensaje:
Hugo ---> hola a todos
hola Hugo
Paco ---> hola Hugo
Escribe tu mensaje:
Luis ---> hola Paco
Luis ---> hola Hugo

C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Luis
Escribe tu mensaje:
Hugo ---> hola a todos
Paco ---> hola Hugo
hola Paco
Luis ---> hola Hugo
Escribe tu mensaje:
hola Hugo
Escribe tu mensaje:
Luis ---> hola Hugo
```

**Figura 3.6.** Luis contesta mensajes.

Paco contesta: “*¿alguien sabe dónde será el concierto?*”.

```
C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Hugo
Escribe tu mensaje:
hola a todos
Hugo ---> hola a todos
Escribe tu mensaje:
Paco ---> hola Hugo
Luis ---> hola Paco
Luis ---> hola Hugo
Paco ---> ¿alguien sabe dónde será el concierto?

C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Paco
Escribe tu mensaje:
Hugo ---> hola a todos
hola Hugo
Paco ---> hola Hugo
Escribe tu mensaje:
Luis ---> hola Paco
Luis ---> hola Hugo
¿alguien sabe dónde será el concierto?
Paco ---> ¿alguien sabe dónde será el concierto?

C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Luis
Escribe tu mensaje:
hola a todos
Paco ---> hola Hugo
hola Paco
Luis ---> hola Hugo
Escribe tu mensaje:
hola Hugo
Escribe tu mensaje:
Luis ---> hola Hugo
Paco ---> ¿alguien sabe dónde será el concierto?
```

Figura 3.7. Paco contesta mensaje.

Hugo contesta: “*será en la plaza central*”.

```
C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Hugo
Escribe tu mensaje:
hola a todos
Hugo ---> hola a todos
Escribe tu mensaje:
Paco ---> hola Hugo
Luis ---> hola Paco
Luis ---> hola Hugo
Paco ---> ¿alguien sabe dónde será el concierto?
será en la plaza central
Hugo ---> será en la plaza central
Escribe tu mensaje:

C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Paco
Escribe tu mensaje:
Hugo ---> hola a todos
hola Hugo
Paco ---> hola Hugo
Escribe tu mensaje:
Luis ---> hola Paco
Luis ---> hola Hugo
¿alguien sabe dónde será el concierto?
Paco ---> ¿alguien sabe dónde será el concierto?
Escribe tu mensaje:
Hugo ---> será en la plaza central

C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Luis
Escribe tu mensaje:
Hugo ---> hola a todos
Paco ---> hola Hugo
hola Paco
Luis ---> hola Hugo
Escribe tu mensaje:
hola Hugo
Escribe tu mensaje:
Luis ---> hola Hugo
Paco ---> ¿alguien sabe dónde será el concierto?
Hugo ---> será en la plaza central
```

Figura 3.8. Hugo contesta mensaje.

### Paco contesta: “¿a qué hora”.

```
C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Hugo
Escribe tu mensaje:
hola a todos
Hugo ---> hola a todos
Escribe tu mensaje:
Paco ---> hola Hugo
Luis ---> hola Paco
Luis ---> hola Hugo
Paco ---> ¿alguien sabe dónde será el concierto?
será en la plaza central
Hugo ---> será en la plaza central
Escribe tu mensaje:
Paco ---> ¿a qué hora?

Administrator: C:\Windows\system32\cmd.exe - java Chat Paco
C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Paco
Escribe tu mensaje:
Hugo ---> hola a todos
hola Hugo
Paco ---> hola Hugo
Escribe tu mensaje:
Luis ---> hola Paco
Luis ---> hola Hugo
¿alguien sabe dónde será el concierto?
Paco ---> ¿alguien sabe dónde será el concierto?
Escribe tu mensaje:
Hugo ---> será en la plaza central
¿a qué hora?
Paco ---> ¿a qué hora?
Escribe tu mensaje:
Administrator: C:\Windows\system32\cmd.exe - java Chat Luis
C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Luis
Escribe tu mensaje:
Hugo ---> hola a todos
Paco ---> hola Hugo
hola Paco
Luis ---> hola Hugo
Escribe tu mensaje:
hola Hugo
Escribe tu mensaje:
Luis ---> hola Hugo
Paco ---> ¿alguien sabe dónde será el concierto?
Hugo ---> será en la plaza central
Paco ---> ¿a qué hora?
```

Figura 3.9. Paco contesta mensaje.

### Luis contesta: “a las 8 PM”.

```
C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Hugo
Escribe tu mensaje:
hola a todos
Hugo ---> hola a todos
Escribe tu mensaje:
Paco ---> hola Hugo
Luis ---> hola Paco
Luis ---> hola Hugo
Paco ---> ¿alguien sabe dónde será el concierto?
será en la plaza central
Hugo ---> será en la plaza central
Escribe tu mensaje:
Paco ---> ¿a qué hora?
Luis ---> a las 8 PM

Administrator: C:\Windows\system32\cmd.exe - java Chat Paco
C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Paco
Escribe tu mensaje:
Hugo ---> hola a todos
hola Hugo
Paco ---> hola Hugo
Escribe tu mensaje:
Luis ---> hola Paco
Luis ---> hola Hugo
¿alguien sabe dónde será el concierto?
Paco ---> ¿alguien sabe dónde será el concierto?
Escribe tu mensaje:
Hugo ---> será en la plaza central
¿a qué hora?
Paco ---> ¿a qué hora?
Escribe tu mensaje:
Luis ---> a las 8 PM

Administrator: C:\Windows\system32\cmd.exe - java Chat Luis
C:\Users\Nodo\Documents>javac Chat.java
C:\Users\Nodo\Documents>java Chat Luis
Escribe tu mensaje:
Hugo ---> hola a todos
Paco ---> hola Hugo
hola Paco
Luis ---> hola Hugo
Escribe tu mensaje:
hola Hugo
Escribe tu mensaje:
Luis ---> hola Hugo
Paco ---> ¿alguien sabe dónde será el concierto?
Hugo ---> será en la plaza central
Paco ---> ¿a qué hora?
a las 8 PM
Escribe tu mensaje:
Luis ---> a las 8 PM
```

Figura 4. Luis contesta mensaje.

### Paco contesta: “gracias, adiós”.

The screenshot shows three separate Windows Command Prompt windows running under Administrator mode. Each window has a title bar indicating it's an administrator session and the command used to run the Java application (Chat.java). The windows are arranged vertically, showing the flow of a conversation between three users: Hugo, Paco, and Luis.

- Top Window (Administrator):** Shows the interaction between Hugo and Paco. Hugo asks about the concert location and time, and Paco responds with "será en la plaza central" and "a las 8 PM". Both end with "gracias, adiós".
- Middle Window (Administrator):** Shows the interaction between Luis and Hugo. Luis asks about the concert location and time, and Hugo responds with "será en la plaza central" and "a las 8 PM". Both end with "gracias, adiós".
- Bottom Window (Administrator):** Shows the interaction between Luis and Paco. Luis asks about the concert location and time, and Paco responds with "será en la plaza central" and "a las 8 PM". Both end with "gracias, adiós".

The system tray at the bottom of the screen shows the date as 3/28/2023 and the time as 6:19 PM. A search bar is visible at the top of the desktop.

Figura 4.1. Paco contesta mensaje.

### Luis contesta: “adiós Paco”.

This screenshot shows the continuation of the conversation from Figure 4.1. The three Windows Command Prompt windows are still open, but the focus is now on Luis' responses.

- Top Window (Administrator):** Luis asks about the concert location and time, and Hugo responds with "será en la plaza central" and "a las 8 PM". Both end with "gracias, adiós".
- Middle Window (Administrator):** Luis asks about the concert location and time, and Hugo responds with "será en la plaza central" and "a las 8 PM". Both end with "gracias, adiós".
- Bottom Window (Administrator):** Luis asks about the concert location and time, and Hugo responds with "será en la plaza central" and "a las 8 PM". Both end with "gracias, adiós".

In the final exchange, Luis sends a message to Paco, which is displayed in the bottom window:

```
Escribe tu mensaje:  
Paco ----> adiós Paco
```

The system tray at the bottom of the screen shows the date as 3/28/2023 and the time as 6:20 PM. A search bar is visible at the top of the desktop.

Figura 4.2. Luis contesta mensaje y finaliza conversación.

## **Conversación sostenida con Chat GPT.**

**Luis Antonio Flores Castro.**

Hola buena noche, ¿Me podrías ayudar con lo siguiente en lenguaje java?

Desarrollar un programa llamado "Chat.java" de tipo multicast.

El programa deberá realizar lo siguiente:

Se deberá pasar como parámetro al programa el nombre del usuario, este se guardará en una variable tipo String llamada "nombre\_usuario".

Se deberá crear un thread el cuál recibirá mensajes y los imprimirá.

Dentro del método main, habrá un ciclo infinito de tipo for.

Dentro del ciclo infinito for, se deberá imprimir lo siguiente "Escribe tu mensaje:" junto de un salto de linea y se leerá el mensaje de tipo String.

Se deberá ingresar un mensaje que se guardará en una variable tipo String llamada "mensaje\_ingresado",

una vez ingresado el mensaje por el usuario, este se deberá enviar a los demás nodos que pertenezcan al grupo que tiene IP igual a 239.0.0.0 a través del puerto 50000.

Este mensaje que se enviará mediante datagrama deberá tener el siguiente formato:

nombre\_usuario ---> mensaje ingresado.

Los mensajes se deberán imprimir de forma correcta si tienen signos de interrogación o palabras con acentos.

No se deberá enviar la longitud del mensaje, se deberá utilizar una longitud fija para el mensaje.

## **Chat GPT.**

¡Claro que sí! A continuación, te dejo el código para el programa Chat.java que cumple con los requerimientos que has indicado:

```
import java.net.*;
import java.io.*;

public class Chat {
    public static void main(String[] args) throws Exception {
        String nombre_usuario = args[0];
        MulticastSocket socket = new MulticastSocket(50000);
        InetAddress grupo = InetAddress.getByName("239.0.0.0");
        socket.joinGroup(grupo);
```

```

Thread thread = new Thread(() -> {
    while (true) {
        byte[] buffer = new byte[1000];
        DatagramPacket mensaje = new DatagramPacket(buffer, buffer.length);
        try {
            socket.receive(mensaje);
        } catch (IOException e) {
            e.printStackTrace();
        }
        String mensaje_recibido = new String(mensaje.getData()).trim();
        System.out.println(mensaje_recibido);
    }
});

thread.start();

BufferedReader teclado = new BufferedReader(new
InputStreamReader(System.in));

for (;;) {
    System.out.print("Escribe tu mensaje: ");
    String mensaje_ingresado = teclado.readLine();
    String mensaje_enviado = nombre_usuario + " ---> " + mensaje_ingresado;
    byte[] buffer = mensaje_enviado.getBytes();
    DatagramPacket mensaje = new DatagramPacket(buffer, buffer.length, grupo,
50000);
    socket.send(mensaje);
}
}
}

```

En el código se utiliza la clase MulticastSocket para crear y unirse a un grupo multicast con la dirección IP 239.0.0.0 y el puerto 50000. También se crea un Thread que recibe mensajes y los imprime en la consola.

Dentro del ciclo infinito for, se lee el mensaje ingresado por el usuario a través del teclado y se envía el mensaje a todos los nodos del grupo multicast utilizando un DatagramPacket.

Finalmente, el mensaje se construye en el formato "nombre\_usuario ---> mensaje ingresado" y se envía a través del socket multicast.

Espero que te sea de ayuda! Si tienes alguna otra pregunta, no dudes en preguntar.

### **Luis Antonio Flores Castro.**

El código que recien generaste muestra el siguiente error: Note: Chat.java uses or overrides a deprecated API. Note: Recompile with -Xlint:deprecation for details. Me parece que es debido a que el método .joinGroup() esta deprecado, creo que se debería utilizar una instancia de NetworkInterface para que funcione, podrías arreglarlo?

### **Chat GPT.**

¡Claro! Tienes razón, el método joinGroup() está deprecado en la versión más reciente de Java. Una alternativa es usar una instancia de NetworkInterface para unirse al grupo multicast. Te dejo el código actualizado con este cambio:

```
import java.net.*;
import java.io.*;

public class Chat {
    public static void main(String[] args) throws Exception {
        String nombre_usuario = args[0];
        MulticastSocket socket = new MulticastSocket(50000);
        InetAddress grupo = InetAddress.getByName("239.0.0.0");
        NetworkInterface ni = NetworkInterface.getByInetAddress(InetAddress.getLocalHost());
        socket.joinGroup(new InetSocketAddress(grupo, 50000), ni);

        Thread thread = new Thread(() -> {
            while (true) {
                byte[] buffer = new byte[1000];
                DatagramPacket mensaje = new DatagramPacket(buffer, buffer.length);
                try {
                    socket.receive(mensaje);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        });
    }
}
```

```
String mensaje_recibido = new String(mensaje.getData()).trim();
System.out.println(mensaje_recibido);
}

});

thread.start();

BufferedReader teclado = new BufferedReader(new
InputStreamReader(System.in));

for (;;) {
    System.out.print("Escribe tu mensaje: ");
    String mensaje_ingresado = teclado.readLine();
    String mensaje_enviado = nombre_usuario + " ---> " + mensaje_ingresado;
    byte[] buffer = mensaje_enviado.getBytes();
    DatagramPacket mensaje = new DatagramPacket(buffer, buffer.length, grupo,
50000);
    socket.send(mensaje);
}
}
```

En este código, se utiliza el método `getByInetAddress()` de la clase `NetworkInterface` para obtener la interfaz de red asociada con la dirección IP local del host. Luego, se utiliza este objeto de interfaz de red para unirse al grupo multicast utilizando el método `joinGroup()`.

Espero que este código actualizado te funcione bien. Si tienes alguna otra pregunta, no dudes en preguntar.

Luis Antonio Flores Castro.

Perfecto, funciona muy bien, ahora dentro de la impresión de "Escribe tu mensaje: ", ¿Podrías añadir un salto de linea por favor?

## Chat GPT.

¡Claro! Es una pequeña modificación, solo debemos agregar un carácter de salto de línea \n al final del mensaje de la línea que imprime "Escribe tu mensaje:". El código quedaría de la siguiente manera:

```
import java.net.*;  
import java.io.*;
```

```

public class Chat {

    public static void main(String[] args) throws Exception {
        String nombre_usuario = args[0];
        MulticastSocket socket = new MulticastSocket(50000);
        InetAddress grupo = InetAddress.getByName("239.0.0.0");
        NetworkInterface ni =
        NetworkInterface.getByInetAddress(InetAddress.getLocalHost());
        socket.joinGroup(new InetSocketAddress(grupo, 50000), ni);

        Thread thread = new Thread(() -> {
            while (true) {
                byte[] buffer = new byte[1000];
                DatagramPacket mensaje = new DatagramPacket(buffer, buffer.length);
                try {
                    socket.receive(mensaje);
                } catch (IOException e) {
                    e.printStackTrace();
                }
                String mensaje_recibido = new String(mensaje.getData()).trim();
                System.out.println(mensaje_recibido);
            }
        });
        thread.start();

        BufferedReader teclado = new BufferedReader(new
        InputStreamReader(System.in));

        for (;;) {
            System.out.print("Escribe tu mensaje:\n"); // agregamos \n al final
            String mensaje_ingresado = teclado.readLine();
            String mensaje_enviado = nombre_usuario + " ---> " + mensaje_ingresado;
            byte[] buffer = mensaje_enviado.getBytes();
            DatagramPacket mensaje = new DatagramPacket(buffer, buffer.length, grupo,
            50000);
            socket.send(mensaje);
        }
    }
}

```

```
    }
}
}
```

Con esta modificación, cada vez que se imprime "Escribe tu mensaje:" en la consola, se imprimirá en una nueva línea. Espero que esto te sea de ayuda.

### Luis Antonio Flores Castro.

Muy bien, ahora como puedo hacer que cuando se envié un mensaje con signos de interrogación o palabras acentuadas, estos caracteres se impriman de forma correcta?

### Chat GPT.

Para que los caracteres con signos de interrogación o acentos se impriman de forma correcta, es importante asegurarse de que la codificación de caracteres utilizada para la entrada y salida de datos sea la misma. En este caso, podemos especificar la codificación de caracteres UTF-8 al crear las instancias del objeto InputStreamReader y OutputStreamWriter, y al convertir los datos del mensaje en un arreglo de bytes para su envío.

El código modificado quedaría de la siguiente manera:

```
import java.net.*;
import java.io.*;

public class Chat {
    public static void main(String[] args) throws Exception {
        String nombre_usuario = args[0];
        MulticastSocket socket = new MulticastSocket(50000);
        InetAddress grupo = InetAddress.getByName("239.0.0.0");
        NetworkInterface ni =
        NetworkInterface.getByInetAddress(InetAddress.getLocalHost());
        socket.joinGroup(new InetSocketAddress(grupo, 50000), ni);

        Thread thread = new Thread(() -> {
            while (true) {
                byte[] buffer = new byte[1000];
                DatagramPacket mensaje = new DatagramPacket(buffer, buffer.length);
                try {
                    socket.receive(mensaje);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        });
    }
}
```

```

        }

        String mensaje_recibido;
        try {
            mensaje_recibido = new String(mensaje.getData(), "UTF-8").trim();
            System.out.println(mensaje_recibido);
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }

});

thread.start();

BufferedReader teclado = new BufferedReader(new InputStreamReader(System.in,
"UTF-8"));

OutputStreamWriter outstream = new OutputStreamWriter(System.out, "UTF-8");

for (;;) {
    outstream.write("Escribe tu mensaje:\n");
    outstream.flush();
    String mensaje_ingresado = teclado.readLine();
    String mensaje_enviado = nombre_usuario + " ---> " + mensaje_ingresado;
    byte[] buffer = mensaje_enviado.getBytes("UTF-8");
    DatagramPacket mensaje = new DatagramPacket(buffer, buffer.length, grupo,
50000);
    socket.send(mensaje);
}
}
}
}

```

Con estas modificaciones, se especifica la codificación de caracteres UTF-8 al crear las instancias de InputStreamReader, OutputStreamWriter y al convertir los datos del mensaje a un arreglo de bytes para su envío.

Espero que esto te sea de ayuda.

**Luis Antonio Flores Castro.**

Ya lo he probado, pero las letras acentuadas y los signos de interrogación no se muestran de forma correcta, ¿Puedes corregirlo?

### **Chat GPT.**

Claro, es posible que la consola en la que estás ejecutando el programa no esté configurada para mostrar caracteres con acentos o signos de interrogación correctamente. Para solucionarlo, podemos intentar establecer la codificación de caracteres adecuada en la consola antes de ejecutar el programa.

chcp 6500.

Para establecer la codificación de caracteres en la consola, podemos ejecutar el siguiente comando en la línea de comandos antes de compilar y ejecutar el programa:

Este comando establece la codificación de caracteres a UTF-8 en la consola de Windows. Si estás utilizando una consola diferente, es posible que necesites utilizar un comando diferente para establecer la codificación de caracteres adecuada.

Una vez que hayas establecido la codificación de caracteres adecuada en la consola, puedes probar el siguiente código, que especifica la codificación de caracteres UTF-8 al crear las instancias de InputStreamReader y OutputStreamWriter, y al convertir los datos del mensaje en un arreglo de bytes para su envío.

### **Luis Antonio Flores Castro.**

Se puede solucionar el problema de la impresión de letras acentuadas y signos de interrogación sin ejecutar el código chcp 65001?

### **Chat GPT.**

Sí, en ese caso puedes intentar establecer la codificación de caracteres adecuada directamente en el código Java. Para hacerlo, puedes especificar la codificación de caracteres al crear las instancias de InputStreamReader y OutputStreamWriter, y al convertir los datos del mensaje en un arreglo de bytes para su envío.

Por ejemplo, puedes modificar el código anterior de la siguiente manera:

```
import java.net.*;
import java.io.*;

public class Chat {
    public static void main(String[] args) throws Exception {
        String nombre_usuario = args[0];
        MulticastSocket socket = new MulticastSocket(50000);
        InetAddress grupo = InetAddress.getByName("239.0.0.0");
        NetworkInterface ni =
        NetworkInterface.getByInetAddress(InetAddress.getLocalHost());
        socket.joinGroup(new InetSocketAddress(grupo, 50000), ni);

        Thread thread = new Thread(() -> {
```

```

        while (true) {
            byte[] buffer = new byte[1000];
            DatagramPacket mensaje = new DatagramPacket(buffer, buffer.length);
            try {
                socket.receive(mensaje);
            } catch (IOException e) {
                e.printStackTrace();
            }
            String mensaje_recibido;
            try {
                mensaje_recibido = new String(mensaje.getData(), "UTF-8").trim();
                System.out.println(mensaje_recibido);
            } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
            }
        }
    });
    thread.start();

    BufferedReader teclado = new BufferedReader(new InputStreamReader(System.in,
    "UTF-8"));
    OutputStreamWriter outstream = new OutputStreamWriter(System.out, "UTF-8");

    for (;;) {
        outstream.write("Escribe tu mensaje:\n");
        outstream.flush();
        String mensaje_ingresado = teclado.readLine();
        String mensaje_enviado = nombre_usuario + " ---> " + mensaje_ingresado;
        byte[] buffer = mensaje_enviado.getBytes("UTF-8");
        DatagramPacket mensaje = new DatagramPacket(buffer, buffer.length, grupo,
50000);
        socket.send(mensaje);
    }
}

```

En este ejemplo, especificamos la codificación de caracteres UTF-8 al crear las instancias de `InputStreamReader` y `OutputStreamWriter`, y al convertir los datos del mensaje en un arreglo de bytes para su envío. Esto debería permitir que se impriman correctamente las letras acentuadas y los signos de interrogación en la consola, sin necesidad de ejecutar el comando `chcp 65001`.

#### **Luis Antonio Flores Castro.**

UTF-8 no me realiza bien la impresión de letras acentuadas y signos de interrogación.

#### **Chat GPT.**

Si UTF-8 no está imprimiendo correctamente los caracteres acentuados y los signos de interrogación, es posible que el problema se deba a que la consola o el terminal que estás utilizando no soporta la codificación UTF-8.

En ese caso, puedes intentar cambiar la codificación utilizada por la consola o el terminal a una que admita los caracteres acentuados y los signos de interrogación. Por ejemplo, si estás utilizando Windows, puedes intentar cambiar la página de códigos de la consola a "Latin-1" o "Windows-1252", que son codificaciones ampliamente utilizadas para caracteres latinos.

Para cambiar la página de códigos de la consola en Windows, puedes seguir estos pasos:

Abre la consola de comandos (`cmd.exe`).

Haz clic con el botón derecho en la barra de título de la ventana de la consola y selecciona "Propiedades".

Selecciona la pestaña "Página de códigos".

En la lista "Página de códigos de consola activa", selecciona la página de códigos que quieras utilizar (por ejemplo, "Latin-1" o "Windows-1252").

Haz clic en "Aceptar" para guardar los cambios.

Una vez cambiada la página de códigos de la consola, deberías poder imprimir correctamente los caracteres acentuados y los signos de interrogación utilizando la codificación correspondiente (en este caso, Latin-1 o Windows-1252).

#### **Luis Antonio Flores Castro.**

Que tipo de codificación utiliza Windows para la impresión de letras acentuadas y signos de interrogación?

#### **Chat GPT.**

Windows utiliza principalmente dos tipos de codificaciones para la impresión de letras acentuadas y signos de interrogación:

1. ANSI: es la codificación predeterminada en versiones antiguas de Windows (por ejemplo, Windows 95, Windows 98 y Windows Me). Esta codificación admite caracteres latinos y algunos caracteres especiales, pero no admite caracteres de otros idiomas (como el japonés o el chino, por ejemplo).

2. UTF-8: es una codificación de caracteres Unicode ampliamente utilizada en versiones más recientes de Windows (por ejemplo, Windows XP, Windows Vista, Windows 7 y Windows 10). Esta codificación admite una amplia gama de caracteres de diferentes idiomas y escrituras, incluyendo letras acentuadas y signos de interrogación.

Es importante tener en cuenta que la codificación utilizada por la consola o el terminal en Windows puede variar según la configuración regional del sistema operativo y la configuración de la aplicación que se esté utilizando. Por lo tanto, es posible que sea necesario ajustar la codificación de la consola o el terminal para que los caracteres acentuados y los signos de interrogación se impriman correctamente.

**Luis Antonio Flores Castro.**

Ya he utilizado UTF-8 para la impresión de letras acentuadas y signos de interrogación pero no funciona.

**Chat GPT.**

Si has utilizado UTF-8 para la impresión de letras acentuadas y signos de interrogación y no funcionó, es posible que debas verificar algunas cosas:

1. Asegúrate de que el archivo fuente de tu programa esté guardado con la codificación UTF-8. Puedes hacer esto en la mayoría de los editores de texto seleccionando la opción de guardar el archivo con codificación UTF-8.
2. Verifica que la consola o el terminal que estás utilizando esté configurado para utilizar la codificación UTF-8. En algunos casos, es posible que debas cambiar la configuración de la consola o el terminal para que utilice la codificación UTF-8. Esto se puede hacer en la configuración de la consola o el terminal, dependiendo del sistema operativo que estés utilizando.
3. Asegúrate de que la fuente que estás utilizando para la consola o el terminal admita los caracteres acentuados y los signos de interrogación. En algunos casos, la fuente predeterminada de la consola o el terminal no admite ciertos caracteres, lo que puede provocar problemas de visualización. En este caso, debes cambiar la fuente de la consola o el terminal a una que admita todos los caracteres que necesitas.

Si después de verificar estas cosas aún tienes problemas con la impresión de letras acentuadas y signos de interrogación, es posible que debas buscar una solución específica para tu situación en particular.

**Luis Antonio Flores Castro.**

¿Puedes modificar el código para que se imprima correctamente las letras acentuadas y signos de interrogación en la terminal de windows?

**Chat GPT.**

Sí, puedes utilizar StandardCharsets.UTF\_8 en lugar de "UTF-8" para imprimir correctamente las letras acentuadas y los signos de interrogación en la terminal de Windows. El código modificado quedaría así:

```

import java.net.*;
import java.io.*;
import java.nio.charset.StandardCharsets;

public class Chat {

    public static void main(String[] args) throws Exception {
        String nombre_usuario = args[0];
        MulticastSocket socket = new MulticastSocket(50000);
        InetAddress grupo = InetAddress.getByName("239.0.0.0");
        NetworkInterface ni = NetworkInterface.getByInetAddress(InetAddress.getLocalHost());
        socket.joinGroup(new InetSocketAddress(grupo, 50000), ni);

        Thread thread = new Thread(() -> {
            while (true) {
                byte[] buffer = new byte[1000];
                DatagramPacket mensaje = new DatagramPacket(buffer,
buffer.length);
                try {
                    socket.receive(mensaje);
                } catch (IOException e) {
                    e.printStackTrace();
                }
                String mensaje_recibido;
                try {
                    mensaje_recibido = new String(mensaje.getData(),
StandardCharsets.UTF_8).trim();
                    System.out.println(mensaje_recibido);
                } catch (UnsupportedEncodingException e) {
                    e.printStackTrace();
                }
            }
        });
        thread.start();
    }
}

```

```

        BufferedReader teclado = new BufferedReader(new
InputStreamReader(System.in, StandardCharsets.UTF_8));

        OutputStreamWriter outstream = new OutputStreamWriter(System.out,
StandardCharsets.UTF_8);

for (;;) {
    outstream.write("Escribe tu mensaje:\n");
    outstream.flush();
    String mensaje_ingresado = teclado.readLine();

    String mensaje_enviado = nombre_usuario + " ---> " +
mensaje_ingresado;
    byte[] buffer = mensaje_enviado.getBytes(StandardCharsets.UTF_8);

    DatagramPacket mensaje = new DatagramPacket(buffer, buffer.length,
grupo, 50000);
    socket.send(mensaje);
}
}
}

```

Espero que esto te ayude a imprimir correctamente las letras acentuadas y los signos de interrogación.

### **Luis Antonio Flores Castro.**

El código recién generado muestra el siguiente error: Chat.java:26: error: exception UnsupportedEncodingException is never thrown in body of corresponding try statement } catch (UnsupportedEncodingException e) { ^ 1 error

### **Chat GPT.**

Tienes razón, el bloque catch para la excepción UnsupportedEncodingException ya no es necesario al utilizar StandardCharsets.UTF\_8, ya que esta excepción es verificada y no puede ser lanzada en tiempo de ejecución. Puedes eliminar ese bloque catch y dejar el código así:

```

import java.net.*;
import java.io.*;
import java.nio.charset.StandardCharsets;

public class Chat {
    public static void main(String[] args) throws Exception {
        String nombre_usuario = args[0];

```

```

        MulticastSocket socket = new MulticastSocket(50000);
        InetAddress grupo = InetAddress.getByName("239.0.0.0");
        NetworkInterface ni =
NetworkInterface.getByInetAddress(InetAddress.getLocalHost());
        socket.joinGroup(new InetSocketAddress(grupo, 50000), ni);

        Thread thread = new Thread(() -> {
            while (true) {
                byte[] buffer = new byte[1000];
                DatagramPacket mensaje = new DatagramPacket(buffer,
buffer.length);
                try {
                    socket.receive(mensaje);
                } catch (IOException e) {
                    e.printStackTrace();
                }
                String mensaje_recibido = new String(mensaje.getData(),
StandardCharsets.UTF_8).trim();
                System.out.println(mensaje_recibido);
            }
        });
        thread.start();

        BufferedReader teclado = new BufferedReader(new
InputStreamReader(System.in, StandardCharsets.UTF_8));
        OutputStreamWriter outstream = new OutputStreamWriter(System.out,
StandardCharsets.UTF_8);

        for (;;) {
            outstream.write("Escribe tu mensaje:\n");
            outstream.flush();
            String mensaje_ingresado = teclado.readLine();
            String mensaje_enviado = nombre_usuario + " ---> " +
mensaje_ingresado;
            byte[] buffer = mensaje_enviado.getBytes(StandardCharsets.UTF_8);
            DatagramPacket mensaje = new DatagramPacket(buffer, buffer.length,
grupo, 50000);

```

```
        socket.send(mensaje);  
    }  
}  
}
```

Este código debería funcionar sin problemas para imprimir letras acentuadas y signos de interrogación en la consola de Windows.

**Luis Antonio Flores Castro.**

Sigue sin imprimir las letras acentuadas y signos de interrogación de forma correcta.

**Chat GPT.**

En ese caso, otra opción que puedes probar es utilizar la codificación "Cp850" en lugar de "UTF-8". Para hacer esto, puedes cambiar la línea:

```
BufferedReader teclado = new BufferedReader(new InputStreamReader(System.in,  
StandardCharsets.UTF_8));
```

a:

```
BufferedReader teclado = new BufferedReader(new InputStreamReader(System.in,  
"Cp850"));
```

y la línea:

```
OutputStreamWriter outstream = new OutputStreamWriter(System.out,  
StandardCharsets.UTF_8);
```

a:

```
OutputStreamWriter outstream = new OutputStreamWriter(System.out, "Cp850");
```

Esto debería permitir que las letras acentuadas y signos de interrogación se impriman correctamente en la terminal de Windows.

**Luis Antonio Flores Castro.**

Muy bien, ahora verificaré el programa

**Chat GPT.**

¡Perfecto! Avísame si tienes alguna otra pregunta o necesitas más ayuda.

**Luis Antonio Flores Castro.**

perfecto, funciona muy bien ahora, Gracias:)

**Chat GPT.**

¡De nada! Me alegro de haber podido ayudarte. No dudes en consultarme si tienes alguna otra duda o problema. Estoy siempre disponible para ayudarte en lo que necesites.

## Código final generado por chat GPT.

```
import java.net.*;
import java.io.*;
import java.nio.charset.StandardCharsets;

public class Chat {
    public static void main(String[] args) throws Exception {
        String nombre_usuario = args[0];
        MulticastSocket socket = new MulticastSocket(50000);
        InetAddress grupo = InetAddress.getByName("239.0.0.0");
        NetworkInterface ni =
NetworkInterface.getByInetAddress(InetAddress.getLocalHost());
        socket.joinGroup(new InetSocketAddress(grupo, 50000), ni);

        Thread thread = new Thread(() -> {
            while (true) {
                byte[] buffer = new byte[1000];
                DatagramPacket mensaje = new DatagramPacket(buffer,
buffer.length);
                try {
                    socket.receive(mensaje);
                } catch (IOException e) {
                    e.printStackTrace();
                }
                String mensaje_recibido = new String(mensaje.getData(),
StandardCharsets.UTF_8).trim();
                System.out.println(mensaje_recibido);
            }
        });
        thread.start();

        BufferedReader teclado = new BufferedReader(new
InputStreamReader(System.in, "Cp850"));
    }
}
```

```
OutputStreamWriter outstream = new OutputStreamWriter(System.out,
"Cp850");

for (;;) {
    outstream.write("Escribe tu mensaje:\n");
    outstream.flush();
    String mensaje_ingresado = teclado.readLine();
    String mensaje_enviado = nombre_usuario + " ---> " +
mensaje_ingresado;
    byte[] buffer = mensaje_enviado.getBytes(StandardCharsets.UTF_8);
    DatagramPacket mensaje = new DatagramPacket(buffer, buffer.length,
grupo, 50000);
    socket.send(mensaje);

}
}
```

## **Conclusiones.**

A través de la realización de esta tarea pude identificar de manera concreta la diferencia entre los programas tradicionales que utilizan TCP y los que utilizan UDP como fue el caso de esta tarea. El funcionamiento de UDP al principio me generaba ciertas dudas ya que respecto al TCP es mucho más simple, debido a que no se tiene que establecer un canal como tal, ni tampoco verificar si algún receptor está listo. Al desarrollar esta tarea en chat GPT, me encontré con diferentes problemas, el principal fue que el chat tomaba el método `.joinGroup()`, la forma en la que se estaba implementando el programa con este método arrojaba errores acerca de la versión ya que esta deprecado en la versión de java que se estaba utilizando, por lo que después de indicarle el error, modificó el código utilizando una instancia de `NetworkInterface` de esta forma funcionó el programa sin problema. Por otro lado, el segundo problema que se tuvo fue la impresión de signos de interrogación y de letras acentuadas, a lo largo de la conversación se utilizaba la codificación “UTF-8”, sin embargo no realizaba la impresión correcta, fue hasta cuando se le realizó la pregunta al chat de que si podría corregir el código de tal forma que las letras acentuadas y signos de interrogación se desplegarán de forma correcta en la terminal de Windows que dio una solución concreta, la codificación que hizo que el programa funcionará de manera exitosa fue la “Cp850”, qué es una codificación que abarca un mayor número de caracteres.

En conclusión, puedo decir que esta tarea fue un poco más sencilla, sin embargo un punto que la hizo fácil fue el nivel de precisión en la redacción del prompt a chat GPT, el ser directo y específico con el facilita mucho las cosas, de esta manera genera respuestas más concretas y con una cantidad de errores menores, ahorrando tiempo.